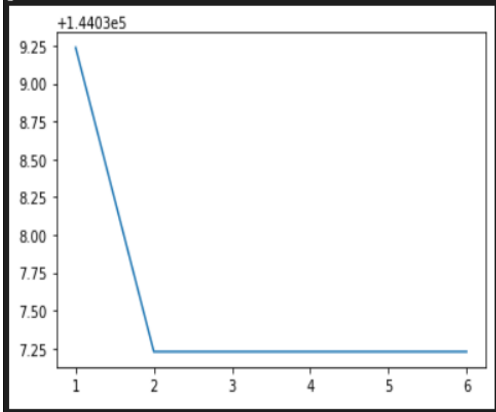
2. a. (i)

For k = 3

Total SSE of each clustering run

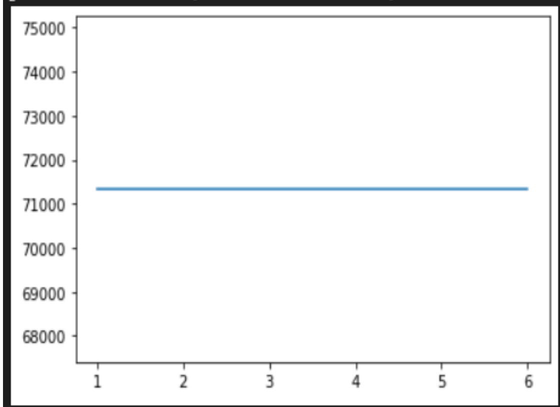
```
3
[144039.23758136347, 144037.22795882105, 144037.22795882105, 144037.22795882105, 144037.22795882105, 144037.22795882105]
```



For k = 5

Total SSE of each clustering run

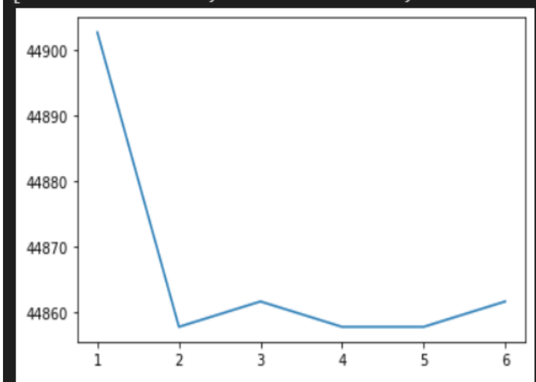
```
5
[71332.5153997308, 71332.5153997308, 71332.5153997308, 71332.5153997308, 71332.5153997308, 71332.5153997308]
```



For k = 7

Total SSE of each clustering run

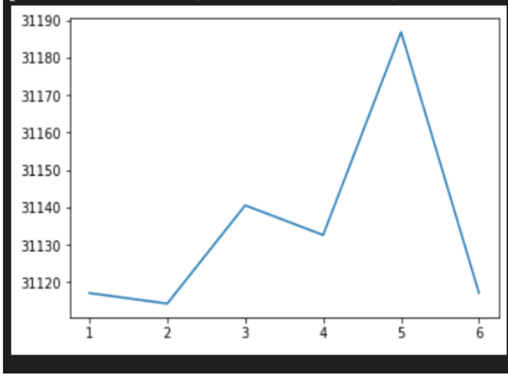
```
7
[44902.731951976595, 44857.75217158104, 44861.63071086091, 44857.75217158104, 44857.75217158104, 44861.63071086091]
```



For k = 9

Total SSE of each clustering run

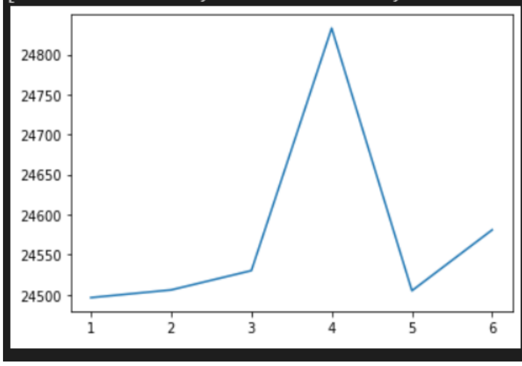
```
9
[31117.093550821737, 31114.275369003553, 31140.51782209972, 31132.587486438344, 31186.80758485895, 31117.093550821737]
```



For k = 11

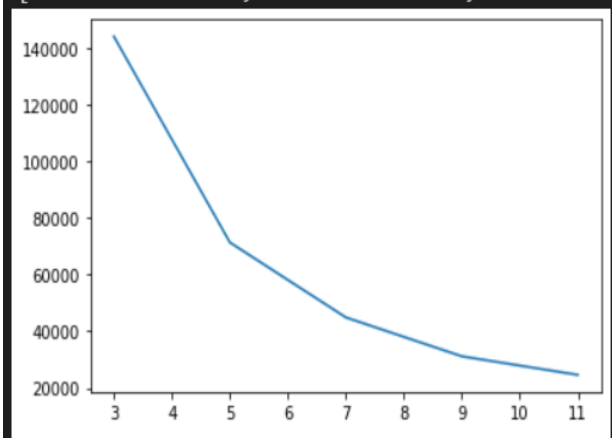
Total SSE of each clustering run

```
11
[24496.286884156354, 24505.87191603465, 24530.009927136045, 24832.78750812723, 24504.87995000337, 24580.88268899966]
```



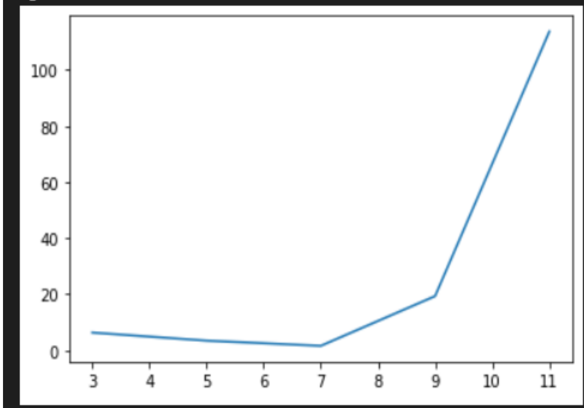
(ii)Average SSE for each k

```
average SSE:
 [144037.22795882105, 71332.85371251618, 44859.045018007666, 31127.086934192368, 24584.618099589858]
```



Standard deviation of SSE for each k

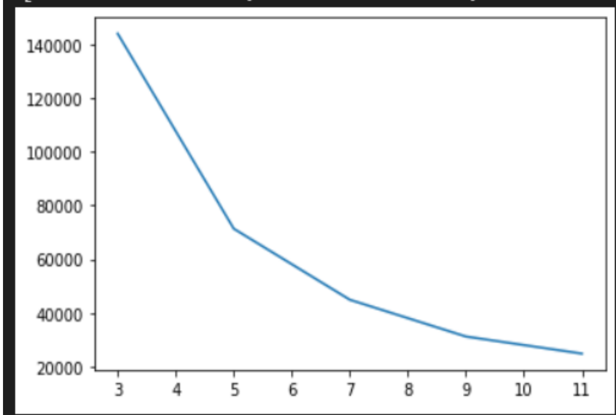
```
std dev of SSE for each k:
 [6.297677097707217, 3.4159507264437656, 1.5673954502974363, 19.312423449152938, 113.75987884365036]
```



(iii) Min and max SSE for each k

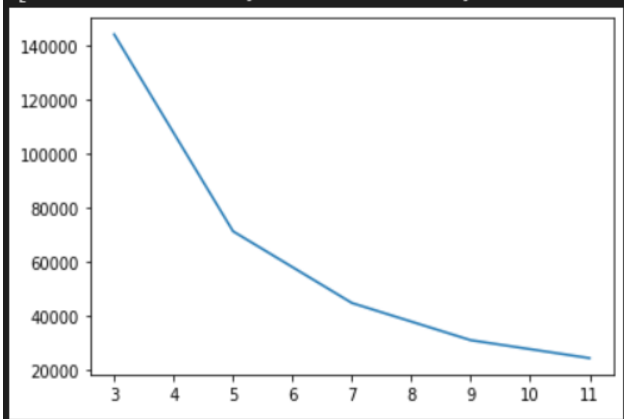
Max SSE for each k

```
max SSE for each k:
 [144039.23758136347, 71341.04890438446, 44857.8551999683, 31156.266077488617, 24789.042556625303]
```

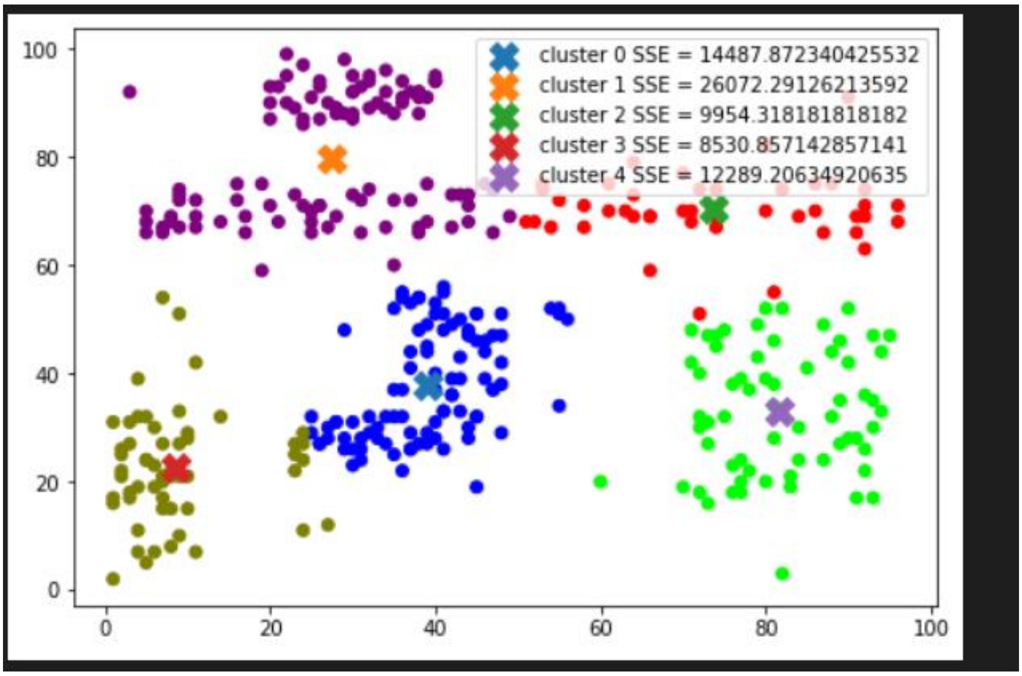


Min SSE for each k

```
min SSE for each k:
 [144037.22795882105, 71332.5153997308, 44857.75217158104, 31114.275369003553, 24492.17944919884]
```



b.



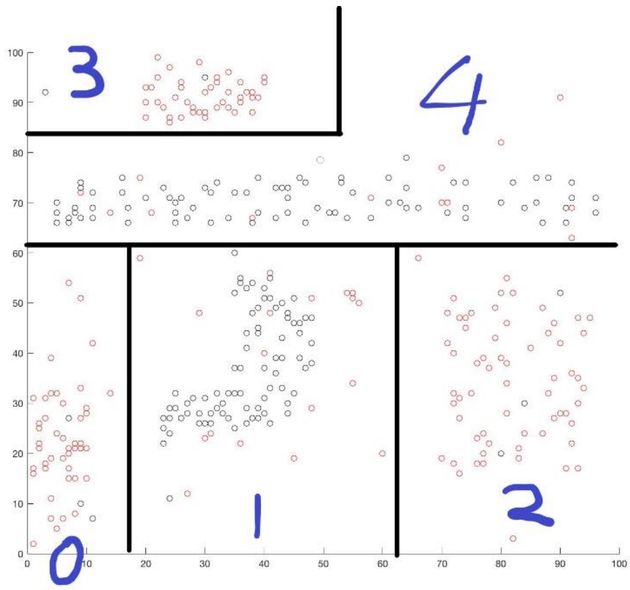
c.

The top cluster should be separated with the belt cluster, and the belt region in the middle should be in a single cluster.

Because the Euclidean distance is used, kmeans cannot correctly cluster the belt region in the middle. This proved that the k means clustering doesn't work well if the data is not spherically distributed.

d.

These data can be classified with linear boundaries, since the margin between each cluster is rathe large.



```python
man_class = []
for i in range(360):
    if df.iloc[i, 0] <= 18 and df.iloc[i, 1] <= 61:
        man_class.append(0)
    elif 18 < df.iloc[i, 0] <= 62 and df.iloc[i, 1] <= 61:
        man_class.append(1)
    elif df.iloc[i, 0] > 62 and df.iloc[i, 1] <= 61:
        man_class.append(2)
    elif df.iloc[i, 0] <= 58 and df.iloc[i, 1] > 84:
        man_class.append(3)
    else:
        man_class.append(4)
m_class = np.array(man_class)

#plt.scatter(df.loc[:, 0], df.loc[:, 1], c = man_class, cmap = "brg")
#plt.show()
```

e.

Construct a contingency matrix, which is symmetric

k_means = cluster

manual clustering = class

nij = intersection(class i, cluster j)

nij = nji

class/cluster	0	1	2	3	4
0	n00	n01	n02	n03	n04
1		n11	n12	n13	n14
2			n22	n23	n24
3				n33	n34
4					n44

rand_idx = (a + d) / (a + b + c + d)

a = sigma(comb(nij, 2)

b = sigma(comb(ni., 2)) - sigma(comb(nij, 2))

c = sigma(comb(n.j, 2)) - sigma(comb(nij, 2))

d = comb(N, 2) - a - b – c

N is the total number of data points

rand index = (a + d) / (a + b + c + d) = 0.8963478799133395

The rand index represents the accuracy of the target clustering, which is kmeans in this case. (Or the similarity between two clustering.)

```python
a = 0
b = 0
c = 0
d = 0

for j in range(5):
    for i in range(5):
        m_ij = m_class[km.labels_ == j]
        nij = m_ij[m_ij == i].size
        if nij >= 2:
            a += spy.comb(nij, 2)

for i in range(5):
    nidot = m_class[m_class == i].size
    if nidot >= 2:
        b += spy.comb(nidot, 2)
b = b - a

for j in range(5):
    ndotj = km.labels_[km.labels_ == j].size
    if ndotj >= 2:
        c += spy.comb(ndotj, 2)
c = c - a

d = spy.comb(360, 2) - a - b - c

randidx = (a + d) / (a + b + c + d)
print(randidx)
```